Approximate Inference Turns Deep Networks into Gaussian Processes

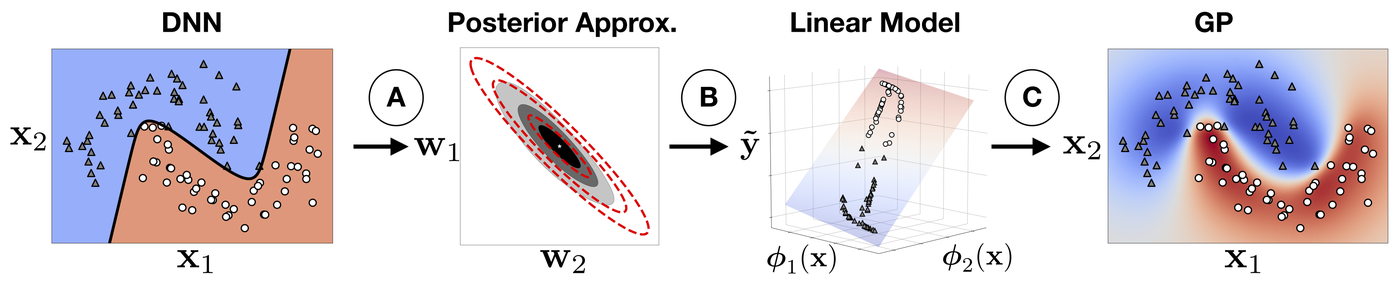
Mohammad Emtiyaz Khan[1], Alexander Immer[2], Ehsan Abedi[2], Maciej Korzepa[3]

[1]Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan
[2]École Polytechnique Fédérale de Lausanne, Switzerland
[3]Technical University of Denmark

Overview

Summary:

- There is limited work and understanding on the connection between training algorithms for deep learning and Gaussian processes
- We want to relate solutions and iterations of a deep-learning algorithm to GP inference
- We show kernels obtained on real datasets and demonstrate the use of the GP marginal likelihood to tune hyperparameters of DNNs



Step A: Find a Gaussian posterior approximation given DNN weights $p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \approx \mathcal{N}(\mathbf{w}|\mu, \mathbf{\Sigma})$

Step B: Find a linear model $\tilde{y} = \phi(\mathbf{x})^\top \mathbf{w} + \epsilon$ with outputs $\tilde{y}$, features $\phi(\mathbf{x})$, and noise ϵ, such that its posterior is equal to the Gaussian approximation, i.e., $p(\mathbf{w}|\tilde{\mathbf{y}}, \mathbf{X}) = \mathcal{N}(\mathbf{w}|\mu, \mathbf{\Sigma})$

Step C: Find a GP whose predictions are the same as those of the linear model.

Step A: Posterior approximation for DNNs

Deep Learning

1. **Training set:** $\mathcal{D} := \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, input $\mathbf{x}_i \in \mathbb{R}^D$, output $\mathbf{y}_i \in \mathbb{R}^K$
2. **Neural network:** $\mathbf{f}_w(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^K$, parameter vector $\mathbf{w} \in \mathbb{R}^P$
3. **Loss function:** $\ell(\mathbf{y}, \mathbf{f}_w(\mathbf{x}))$ twice diff. and strictly convex in $\mathbf{f}_w(\mathbf{x})$

$$\mathbf{w}_* = \arg\min_w \bar{\ell}(\mathcal{D}, \mathbf{w}) := \sum_{i=1}^N \ell(\mathbf{y}_i, \mathbf{f}_w(\mathbf{x}_i)) + \tfrac{1}{2}\delta \mathbf{w}^\top \mathbf{w},$$

We define the Jacobian $\mathbf{J}_w(\mathbf{x}) := \nabla_w \mathbf{f}_w(\mathbf{x})^\top$, the loss residual $\mathbf{r}_w(\mathbf{x}, \mathbf{y}) := \nabla_f \ell(\mathbf{y}, \mathbf{f}_w(\mathbf{x}))$ and the loss Hessian $\mathbf{\Lambda}_w(\mathbf{x}, \mathbf{y}) := \nabla^2_{ff} \ell(\mathbf{y}, \mathbf{f}_w(\mathbf{x}))$.

Approximations to the Posterior

Laplace GGN approximation $q_{\text{Lap}}(\mathbf{w}) := \mathcal{N}(\mathbf{w}|\mathbf{w}_*, \widetilde{\mathbf{\Sigma}})$ where

$$\widetilde{\mathbf{\Sigma}}^{-1} = \sum_{i=1}^N \mathbf{J}_{w_*}(\mathbf{x}_i)^\top \mathbf{\Lambda}_{w_*}(\mathbf{x}_i, \mathbf{y}_i) \mathbf{J}_{w_*}(\mathbf{x}_i) + \delta \mathbf{I}_P$$

Online GGN iterates $q_t(\mathbf{w}) := \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t)$ with $\boldsymbol{\mu} := \boldsymbol{\mu}_t$ below

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu} - \beta_t \mathbf{\Sigma}_{t+1} \left[\sum_{i=1}^N \mathbf{J}_\mu(\mathbf{x}_i)^\top \mathbf{r}_\mu(\mathbf{x}_i, \mathbf{y}_i) + \delta \boldsymbol{\mu} \right],$$

$$\mathbf{\Sigma}_{t+1}^{-1} = (1 - \beta_t)\mathbf{\Sigma}_t^{-1} + \beta_t \left[\sum_{i=1}^N \left[\mathbf{J}_\mu(\mathbf{x}_i)^\top \mathbf{\Lambda}_\mu(\mathbf{x}_i, \mathbf{y}_i) \mathbf{J}_\mu(\mathbf{x}_i) \right] + \delta \mathbf{I}_P \right].$$

Step B: DNN Inference to Linear Model

Define a linear model with prior $\mathbf{w} \sim \mathcal{N}(0, \delta^{-1}\mathbf{I}_P)$, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, (\mathbf{\Lambda}_{w_*}(\mathbf{x}, \mathbf{y}))^{-1})$, and dataset as shown below:

$$\begin{aligned} \text{Model:} \quad & \tilde{\mathbf{y}} = \mathbf{J}_{w_*}(\mathbf{x})\mathbf{w} + \boldsymbol{\epsilon}, \\ \text{Data } \widetilde{\mathcal{D}}\text{:} \quad & \tilde{\mathbf{y}}_i := \mathbf{J}_{w_*}(\mathbf{x}_i)\mathbf{w}_* - (\mathbf{\Lambda}_{w_*}(\mathbf{x}_i, \mathbf{y}_i))^{-1} \mathbf{r}_{w_*}(\mathbf{x}_i, \mathbf{y}_i). \end{aligned}$$

The Laplace GGN posterior approximation of the deep network is *equivalent* to the exact posterior distribution of this linear model, i.e.

$$q_{\text{Lap}}(\mathbf{w}) = p(\mathbf{w}|\widetilde{\mathcal{D}})$$

Define a linear model with $\mathbf{w} \sim \mathcal{N}(\mathbf{m}_t, \mathbf{S}_t)$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, (\beta_t \mathbf{\Lambda}_\mu(\mathbf{x}, \mathbf{y}))^{-1})$ where $\boldsymbol{\mu} := \boldsymbol{\mu}_t$, $\mathbf{S}_t := ((1 - \beta_t)\mathbf{\Sigma}_t^{-1} + \beta_t \delta \mathbf{I}_P)^{-1}$, $\mathbf{m}_t := (1 - \beta_t)\mathbf{S}_t \mathbf{\Sigma}_t^{-1} \boldsymbol{\mu}_t$:

$$\begin{aligned} \text{Model:} \quad & \tilde{\mathbf{y}}_t = \mathbf{J}_\mu(\mathbf{x})\mathbf{w} + \boldsymbol{\epsilon} \\ \text{Data } \widetilde{\mathcal{D}}_t\text{:} \quad & \tilde{\mathbf{y}}_{t,i} := \mathbf{J}_\mu(\mathbf{x}_i)\boldsymbol{\mu} - (\mathbf{\Lambda}_\mu(\mathbf{x}_i, \mathbf{y}_i))^{-1} \mathbf{r}_\mu(\mathbf{x}_i, \mathbf{y}_i). \end{aligned}$$

The *Online GGN* updates to the posterior approximation of the deep network are equivalent to exact inference in this linear model, i.e.

$$q_{t+1}(\mathbf{w}) = p(\mathbf{w}|\widetilde{\mathcal{D}}_t)$$

Step C: From Linear Model to GP

Using the equivalence between the weight-space view and the function-space view, we get a GP regression model whose posterior predictive distribution $p(f(\mathbf{x}_*)|\mathbf{x}_*, \widetilde{\mathcal{D}})$ is equal to that of the linear model.

For the Laplace GGN posterior approximation:

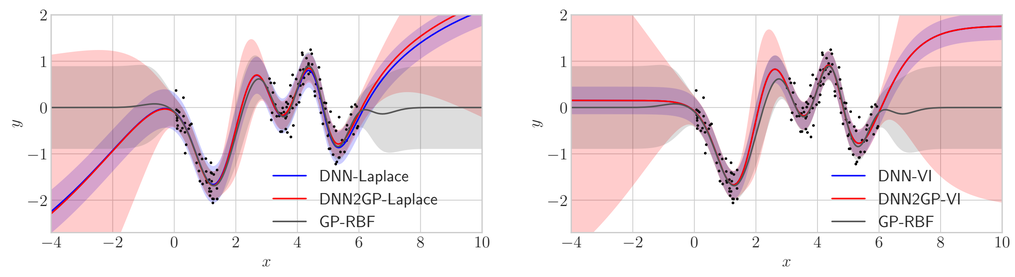
$$\tilde{\mathbf{y}} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\epsilon}, \text{ with } \mathbf{f}(\mathbf{x}) \sim \mathcal{GP}\left(0, \delta^{-1}\mathbf{J}_*(\mathbf{x})\mathbf{J}_*(\mathbf{x}')^\top\right).$$

For the Online GGN updates:

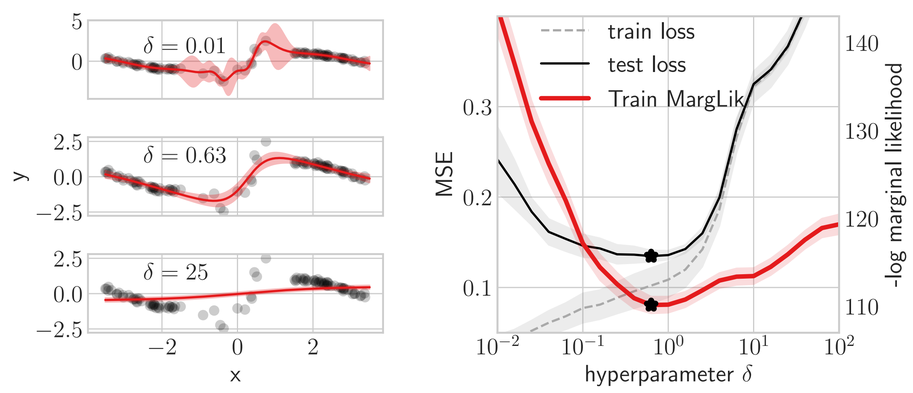
$$\tilde{\mathbf{y}}_t = \mathbf{f}_t(\mathbf{x}) + \boldsymbol{\epsilon}, \text{ with } \mathbf{f}_t(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{J}_\mu(\mathbf{x})\mathbf{m}_t, \mathbf{J}_\mu(\mathbf{x})\mathbf{S}_t\mathbf{J}_\mu(\mathbf{x}')^\top\right).$$

Applications

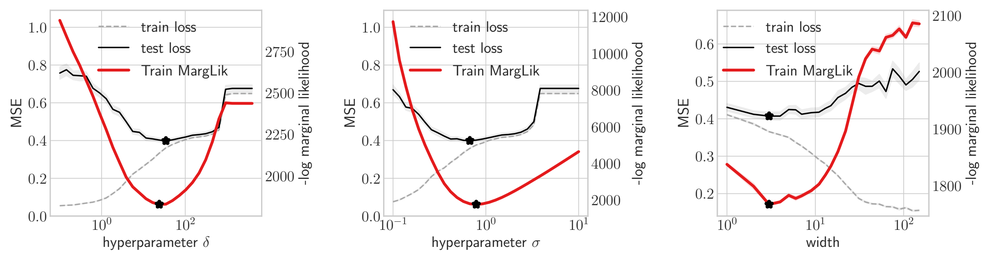
Uncertainty Estimation: by converting a neural network to a GP, we can analytically obtain high quality predictive uncertainty estimates. Below, predictive uncertainty of DNN2GP for Laplace (left) and VI (right) diagonal Gaussian approximation on a toy regression dataset:



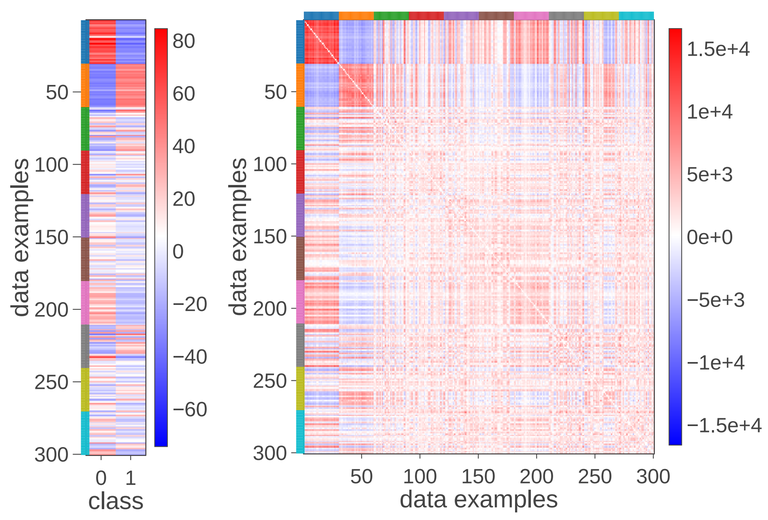
Model Selection via GP Marginal Likelihood: marginal likelihood of the GP allows Bayesian model selection for deep learning using only training data. For example, selection of δ on sinusoidal toy data:



On UCI Wine data for δ, observation noise variance σ^2, network width:



Generalization and Interpretability: posterior mean and GP kernel on 2 out of 10 MNIST classes. We clearly see that the *in-class* digits are assigned higher posterior mean and the correlation learned is realistic.



Possible future applications: online hyperparameter tuning, coreset selection, continual learning, active learning, etc.

References

[1] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. SIAM Review, 2018.

[2] Arthur Jacot, Gabriel Franck, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. NIPS, 2018.

[3] Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. ICML, 2018.